Manmeet Singh Hayer

Senior Project Proposal

*Finvestor*

Section: Maria Webb

Sept 7, 2025

**Table of Contents**

[**Project Summary 3**](#_heading=h.d7p84ranue9f)

[**Significance 4**](#_heading=h.toy0fxn9a7sm)

[**Required Tools & Availability 5**](#_heading=h.2xmk2mv6ofsn)

[**Demonstration Plans 6**](#_heading=h.hsdh4dvughlu)

[**Qualifications 6**](#_heading=h.apnigtc5fuh)

[**Project Specifications 8**](#_heading=h.7l4twpybukhr)

[**Functional Specifications 8**](#_heading=h.d4dwps5q86bb)

[**User Interface Specification 10**](#_heading=h.5xe2oc4671lx)

[**Technical Details 13**](#_heading=h.bt9q2xwm6mdb)

[**Developer Work Plan 15**](#_heading=h.l3r98j7kkr5m)

[**Timeline 16**](#_heading=h.v2syqf7g0qek)

[**∙**](#_heading=h.j9j2yibr3wpd) **Features: Home dashboard with ticker list, basic chart functionality with time ranges, navigation structure for all main pages, responsive design. 16**

[**∙**](#_heading=h.bnwl4os6pd8f) **Demo: Navigate home, swap tickers to update chart, demonstrate time range selection, show navigation to stub pages. 17**

[**∙**](#_heading=h.s3sbwo4306ip) **Acceptance: Ticker navigation smooth, charts render correctly for time ranges, design responsive, nav links functional. 17**

[**Checkpoint #2 (Oct 9) 17**](#_heading=h.7p1jq8ihxrtn)

[**∙**](#_heading=h.xiop37509qy4) **Features: Watchlist CRUD, portfolio creation with inception dates, multi-year OHLCV candlesticks with volume pane, fundamentals (P/E, market cap). 17**

[**∙**](#_heading=h.q1mzhmgxta27) **Demo: Create a watchlist, add tickers, create a portfolio, show ticker chart + volume + fundamentals. 17**

[**∙**](#_heading=h.vpfk4mgvztc3) **Acceptance: Watchlists persist, portfolios store inception, charts accurate, fundamentals display when available. 17**

[**Checkpoint #3 (Oct 30) 18**](#_heading=h.vpfk4mgvztc3)

[**∙**](#_heading=h.38jh22daylpv) **Features: Paper trading with order placement + positions, portfolio analytics (CAGR, vol, max drawdown), SMA/EMA/RSI overlays, benchmark comparisons. 18**

[**∙**](#_heading=h.pcy2zrz5830f) **Demo: Place a buy order and show updates, display analytics tiles, toggle indicators, add benchmark overlay. 18**

[**∙**](#_heading=h.qvjmx80e4vs) **Acceptance: Trading updates positions/cash correctly, analytics calculations accurate, indicators render, benchmark overlays work, transaction history logs. 18**

[**Checkpoint #4 (Nov 20) 18**](#_heading=h.qvjmx80e4vs)

[**∙**](#_heading=h.ispcsiad14h5) **Features: Risk metrics (Sharpe, preliminary VaR), 1-minute intraday charts (last 7 days), complete Methods page. 18**

[**∙**](#_heading=h.vvxr85a16rfp) **Demo: Portfolio risk view with Sharpe/VaR, intraday tab with auto-refresh and update timestamp, Methods page with sources/formulas. 18**

[**∙**](#_heading=h.1c4a8l85xtz4) **Acceptance: Risk metrics calculated, intraday charts update correctly (with limitations noted), Methods page clear and complete, all features integrated smoothly. 18**

[**Future Enhancements 18**](#_heading=h.bl7tq1keul4m)

[**Ethical Considerations 19**](#_heading=h.7z1xfzl3uvkp)

[**Bibliography 20**](#_heading=h.iy6jr048hqn8)

# Project Summary

Finvestor is a web application designed to make investing concepts less intimidating by giving students and beginners a safe place to experiment. The platform allows users to search U.S. equities, build watchlists, view historical and intraday candlestick charts, and simulate trades using paper portfolios. Beyond raw price charts, Finvestor highlights fundamentals like P/E ratios and market cap, and provides key risk metrics such as volatility, drawdown, and Sharpe ratios. All data comes from free sources like Yahoo Finance and FRED, making the app accessible at no cost.

The project’s academic purpose is twofold: it applies computer science techniques—web frameworks, APIs, and databases—to real-world finance problems, and it creates a teaching tool for financial literacy. For the student developer, Finvestor offers an opportunity to connect coursework in Finance and Computer Science while addressing a broader need for approachable, transparent investing education. Ultimately, this project functions as both a working application and a demonstration of how software can lower barriers to financial knowledge.

# Significance

Modern investing is profoundly driven by computer science – from algorithmic trading systems to scrutinizing data-analysis networks – making technical literacy increasingly important in the financial world. “Algorithmic trading generates between 60 and 75 percent of the total trading activity on the U.S. stock exchange and many other established financial markets”, and around 40% in developing nations like India(Prachi). This integration of CS in finance means that understanding coding, data structures, and automation is now an integral part of understanding how the market moves. **Finvestor’s** significance lies partly in bringing some of these advanced concepts down to an approachable level. By using *software* to simulate trading decisions derived from technical analysis, the project concretely demonstrates how computational tools can inform investment decisions – for example, automatically calculating risk metrics or visualizing real-time price changes for a potential buy.

Additionally, the project addresses the accessibility of financial tools and education. In recent years, there has been a surge of new retail investors entering the stock market, especially facilitated by easy-to-use mobile apps. In fact, by 2021 individual retail traders comprised about 25% of U.S. equity trading volume (nearly double the share from a decade earlier) (Einhorn). Despite this increased participation, many retail investors lack access to the sophisticated tools and insights that professionals have. Often, high-quality financial data platforms and analytics are locked behind expensive subscriptions or paywalls – for instance, professional terminals can cost tens of thousands of dollars per year (Gratton). This creates an information gap between those with resources & wealth and those without. Finvestor protests this in its aim to *democratize equity market education*: it leverages free and open data (Yahoo Finance, FRED) and presents it in an interactive format at no cost to the user. By doing so, it helps lower the barrier to entry for students and beginners who want to learn about investing, along with making it a much easier to follow process. Anyone with a web browser can practice trading and see risk/return metrics that normally might require expensive software or deep Excel know-how.  
This project is academically significant for me as a student developer in finance and perfectly embodies DePauw’s emphasis on interdisciplinary learning. I’m applying mathematical concepts using software programming to directly solve problems in business & finance. I see this as a steppingstone towards developing more sophisticated quantitative models or even time-sensitive trading strategies in the future. From a societal perspective, Finvestor aligns with the movement to improve financial literacy through technology by giving beginners a risk-free, interactive environment to build confidence and avoid common pitfalls. Going forward, I envision building on this project by contributing open-source resources, creating widely accessible financial datasets, and potentially launching an educational nonprofit to help under-resourced communities learn about personal finance, saving and investing, managing debt, and understanding risk. In this way, Finvestor is not only relevant to my academic development but also has the potential to scale into a socially impactful tool that broadens financial knowledge and participation.

# Required Tools & Availability

I will develop Finvestor on my MacBook running macOS 10.15.7 (Catalina). My setup includes Visual Studio Code (VS Code) v1.81.1 for editing and Git/GitHub for version control. I’ll use Python 3.11 with FastAPI on the back end, Node.js v22.19.0 (LTS) with React 18, Vite v5.4.20 and Tailwind CSS on the front end, and PostgreSQL 15.14 locally for persistence (accessed with psql). For charts I’ll use Apache ECharts (v5.4), and for data I’ll use yfinance (v0.2.27) and the FRED API (v0.4.5) so the app stays free and transparent. I’ll sketch low-fi screens and component notes in Figma and translate those into React components. All tools are free or student-accessible and are installed or easy to install on this Mac.

# Demonstration Plans

Demonstration of Finvestor will be done in the Linux Lab. To ensure that everything works properly, I will practice my demonstration ahead of time. I will plan to join our Zoom class meeting to screen share with my laptop as the preferred method (the link to join will be sent by the professor ahead of time), but I will also have adapters required to display my Mac on the projector if needed. In case of unexpected issues, I will record my presentation ahead of time and either present the video on the zoom meeting or share the recording with the professor.

# Qualifications

My background combines research discipline, web development, and financial data analysis, giving me the toolkit to bring Finvestor from proposal to reality. At DePauw, I worked with Professor Jeff Hansen on a chemistry research project where I built the front end for an interactive e-book using Vue.js, Bootstrap, and CSS. The work was focused on building responsive layouts, styling interactive modules, and making sure pages worked consistently across devices. While the content was in chemistry, the technical work gave me practice in structuring web applications with established frameworks and refining user interfaces for clarity.

In addition, I have developed and deployed my own [personal website](https://personal-website-wine-two-90.vercel.app/), hosted on Vercel and maintained on GitHub. That project was built with a modern React 18 stack using TypeScript and Vite, and included features such as React Router v6 for navigation, Framer Motion for scroll-based animations, and a Tailwind-based design system with responsive layouts and dark mode. I also set it up with a JSON-based content file so that projects and experience could be updated in one place and reflected across the site. Performance optimizations like code-splitting, lazy loading, and image optimization helped the site load quickly and remain smooth to navigate. Working through these implementation details taught me a lot about structuring components, debugging issues, and keeping the codebase manageable.

I also built a very basic financial dashboard that focused solely on S&P 500 data over the past couple of decades, specifically looking at stock price movements around earnings announcements. This project used a PostgreSQL database with a three-table schema (tickers, earnings events, daily prices), managed through a Python FastAPI backend with SQLAlchemy ORM. While the scope of that project was limited, it gave me exposure to working with historical financial data, setting up queries to extract useful time windows, and validating that results matched expectations. It served as a helpful introduction to handling market data but is far narrower in focus than what Finvestor aims to accomplish.

During my internship at ABM Industries, I worked on the Finance team where I learned enterprise platforms such as Oracle EPM, Anaplan, and Power BI on a tight timeline. A key part of my role involved migrating and optimizing SQL queries to ensure compatibility and performance in Azure SQL environments. This experience built my confidence in troubleshooting data pipelines, improving query efficiency, and adapting to new technical systems quickly—all skills that will be directly relevant when managing Finvestor’s PostgreSQL backend and APIs.

Together, these experiences—front-end development for a research project, building a personal portfolio site with modern frameworks, a focused financial dashboard, and SQL optimization in a professional setting—show that I can handle both the technical and domain challenges of Finvestor. They demonstrate that I can design intuitive user interfaces, manage and process market data accurately, and follow through from concept to working deployment. Combined with my coursework in Finance and Computer Science, I am confident in my ability to deliver a robust and credible application.

# Project Specifications

## Functional Specifications

**Search & Watchlists**

* Autocomplete ticker search (major US equities) with symbol, name, and exchange.
* Add/remove tickers to multiple named watchlists with instant updates.
* Watchlist tiles show last price, $/% daily change, and click-through to ticker pages.
* Organize watchlists with create/rename/delete and drag-and-drop reorder.

**Ticker Page (Historical Analysis)**

* Multi-year daily OHLCV **candlestick charts** with range selectors (1M, 3M, 6M, 1Y, 5Y, MAX).
* Indicator overlays: SMA, EMA, RSI (with toggles & period controls).
* Volume pane below chart with average volume overlay.
* Basic fundamentals (P/E, market cap, volume, and other financials available).
* Benchmark overlays: SPY, QQQ, DIA.

**Intraday Snapshot**

* 1-minute candles covering the last 7 days.
* Auto-refresh tiles (current price, $/%) updating every 30–60 seconds.
* Volume pane synced with intraday chart.

**Paper Trading**

* Market buy/sell orders with quantity, validation, est. cost.
* Updates positions, cash, cost basis, and P&L.
* Transaction history with timestamp, ticker, qty, price, total.
* Prevents invalid trades (no cash, zero qty, off-hours).

**Portfolios & Analytics**

* Create multiple portfolios (custom name, cash- restricted to an initial amount).
* Positions table- ticker, qty, avg cost, current value, unrealized P&L, % return.
* Metrics: cumulative return, CAGR, volatility, max drawdown, Sharpe ratio (using FRED).
* Compare up to 3 portfolios with normalized curves and side-by-side metrics.

**Methods & Transparency**

* Data sources: yfinance (prices), FRED (risk-free).
* Clear formula documentation for metrics & indicators.
* Assumptions + known limitations (data gaps, free API quirks).

**Non-Goals**

* Real money trades, options/derivatives, deep intraday beyond 7 days, margin/short.

## User Interface Specification

**Global**

* Responsive layout with a left sidebar navigation (links: Home, Ticker, Watchlists, Portfolios, Compare, Methods) and a top header containing global search and user menu.
* Consistent design tokens: font sizes (text-xs through text-4xl), spacing (space-1 to space-16), border radius (rounded-sm to rounded-2xl), and shadow levels (shadow-sm to shadow-2xl).
* Interaction states: hover highlighting (e.g., hover:bg-gray-50), focus indicators (focus:ring-2 focus:ring-blue-500), disabled styles (disabled:opacity-50), and skeleton loaders with shimmer animations while data loads.
* Notification system: toast messages appear for trade confirmations, error alerts, and system updates with both timed auto-dismiss and manual dismissal.

**Home Dashboard**

* Top section with a **ticker search bar** (autocomplete) and a **market overview** panel showing major indices (S&P 500, NASDAQ, Dow) with current values and daily changes.
* Watchlist preview showing top 6–8 tickers from the user’s primary watchlist, each with price tile, dollar and percentage change indicators, and “View All” link.
* Tips panel featuring short rotating notes on investing concepts with links to relevant Methods page entries. (OPEN)

**Ticker Page**

* Header row: ticker symbol, company name, exchange, current price, daily change ($ and %), color-coded gain/loss, and an “Add to Watchlist” button reflecting current state.
* Chart area: OHLCV **candlestick chart** with selectable overlays (SMA, EMA, RSI), synchronized volume pane, and time range chips (1M, 3M, 6M, 1Y, 5Y, MAX).
* Fundamentals strip: simple metrics (P/E, market cap, etc.) presented horizontally with placeholders if unavailable.
* Benchmark indice overlay toggles (SPY, QQQ, DIA) with color-coded lines and legend.
* Intraday tab: 1-minute candlestick chart for the last 7 days with an “Updated at [time]” indicator.
* Trade ticket drawer: slides from bottom, inputs for quantity and order type, estimated cost display, and validation messages for invalid entries.

**Watchlists**

* Two view modes: detailed table view and compact card view with sortable columns (ticker, price, % change, volume).
* Create/edit modal with form validation for watchlist name and ticker search integration.

**Portfolios**

* Overview panel: small equity curve chart and key metric tiles in a responsive grid designed using Tailwind utility classes.
* Positions table: ticker, quantity, average cost, current price, market value, unrealized P&L, % return, with sortable columns.
* Transactions table: chronological trade history with expandable rows for details, search filter, and export option.
* Prominent “Place Trade” button opening the trade drawer, pre-filled with the portfolio context.

**Compare**

* Portfolio selector: choose up to 3 portfolios for side-by-side comparison.
* Metric comparison grid: normalized metrics (CAGR, volatility, Sharpe, max drawdown) with color-coded highlights.
* Normalized equity curves chart with distinct colors and interactive legend.
* Time range selector (1M, 3M, 6M, 1Y, 2Y, 5Y, MAX) with automatic chart update.

**Methods**

* Collapsible accordion sections: Data Sources, Formulas, Assumptions, Limitations.
* Mathematical notation displayed alongside plain explanations and small worked examples.
* Notes on data quality: update frequency, known gaps, and reliability.

**Accessibility**

* Full keyboard navigation: logical tab order, arrow navigation for menus, keyboard shortcuts for frequent actions.
* Skip links to jump directly to main content.
* Focus management in modals/drawers (focus trap + restore).
* Empty states with helpful text and call-to-action buttons (e.g., “No trades yet – place your first trade”).
* Error states with plain-language messages, retry option, and fallback content.

# Technical Details

**Architecture**

* Single-page application frontend: React 18 with React Router for routing and Tailwind CSS for styling.
* Backend: FastAPI (Python) with Pydantic models for validation.
* Database: PostgreSQL installed natively on macOS via Postgres.app.
* Data sources: yfinance for daily and intraday stock prices; FRED API for risk-free rates.
* Intraday updates: polling every 30–60 seconds.

**Data Model**

* users: UUID primary key, email login, and preferences.
* tickers: symbol, name, exchange metadata.
* watchlists: many-to-many between users and tickers.
* portfolios: name, inception date, initial cash, fk:user.
* orders: order id, ticker, qty, type, price, timestamp, status.
* fills: record of executed trades linked to orders.
* positions: derived from fills (ticker, qty, avg cost, market value).
* price\_series: OHLCV data for daily and 1-min, indexed by (ticker, date).
* risk\_free\_series: cached FRED data, indexed by date.

**Algorithms**

* Returns: log returns, cumulative returns, CAGR with correct period adjustments.
* Volatility: annualized standard deviation of returns.
* Max drawdown: rolling maximum with drawdown percentage. (OPEN)
* Sharpe ratio: excess return divided by volatility, using FRED risk-free.
* Indicators: SMA (user-configurable periods), EMA, RSI (14-period default).
* Benchmarks: normalized to portfolio inception date.

**Backend Implementation**

* FastAPI routers separated by domain (auth, portfolios, watchlists, data).
* Session-based authentication for dev.
* Async database operations using SQLAlchemy with connection pooling.
* Background tasks (FastAPI BackgroundTasks or cron job) for daily price refresh.

**Frontend Implementation**

* React Router with lazy loading and error boundaries.
* State management with React Context for global entities (user, portfolios) and custom hooks for fetching.
* ECharts for candlestick/line charts with volume pane and overlays.
* Component library: reusable elements (Button, Modal, Table, Chart) styled with Tailwind.
* React Hook Form for controlled inputs and validation.

**Development Environment**

* Postgres installed locally from Postgres.app.
* Backend run with Uvicorn (uvicorn app.main:app --reload).
* Frontend run with Vite dev server (npm run dev).
* Environment variables via .env.

**Tradeoffs**

* Free APIs → limited depth and reliability; acceptable for educational use.
* Polling instead of streaming → simple to implement and stable, with the option to upgrade later.
* Authentication simplified (sessions) to prioritize core trading features.

# Developer Work Plan

* **Week of Sept 11: Project Setup & Foundation**

Objectives: Set up development environment, create project structure, establish initial database schema. [4.5 hours]

Deliverables: React app with routing, FastAPI backend with starter endpoints, PostgreSQL 15 database with base tables.

* **Week of Sept 18: Core Data Infrastructure**

Objectives: Implement yfinance integration, create ticker search functionality, build core data models. [6 hours]

Deliverables: Working ticker search with autocomplete, price data fetching, basic data validation.

* **Week of Sept 25: Checkpoint 1 Preparation**

Objectives: Build home dashboard, implement ticker list display, create basic chart functionality. [7 hours]

Deliverables: Home page with sample S&P 500 ticker list, working chart swap functionality, navigation structure.

* **Week of Oct 2: Checkpoint 1 Demo & Watchlist Foundation**

Objectives: Present Checkpoint 1, begin watchlist functionality, start portfolio creation. [6.5 hours]

Deliverables: Checkpoint 1 presentation, watchlist data models, basic portfolio UI.

* **Week of Oct 9: Watchlist & Portfolio Implementation**

Objectives: Complete watchlist CRUD operations, build portfolio creation flow, implement simple user authentication (session-based). [8 hours]

Deliverables: Full watchlist management, portfolio creation with inception dates, basic login sessions.

* **Week of Oct 16: Historical Charts & Fundamentals**

Objectives: Implement multi-year OHLCV candlestick charts, add volume pane, integrate fundamentals (P/E, market cap). [7.5 hours]

Deliverables: Configurable time range charts, volume analysis, display of fundamentals when available.

* **Week of Oct 23: Checkpoint 2 Demo & Trading Foundation**

Objectives: Present Checkpoint 2, begin paper trading system, start portfolio analytics. [7 hours]

Deliverables: Checkpoint 2 presentation, order placement UI, basic position tracking.

* **Week of Oct 30: Paper Trading Implementation**

Objectives: Complete order placement system, implement position tracking, build transaction history. [8.5 hours]

Deliverables: Full paper trading flow, real-time position updates, transaction logging.

* **Week of Nov 6: Analytics & Technical Indicators**

Objectives: Implement portfolio analytics (CAGR, volatility, max drawdown), add SMA/EMA/RSI overlays, benchmark comparison. [9 hours]

Deliverables: Analytics metric tiles, technical indicators on charts, benchmark overlay toggles.

* **Week of Nov 13: Checkpoint 3 Demo & Risk Metrics**

Objectives: Present Checkpoint 3, begin risk calculations, start intraday functionality. [8 hours]

Deliverables: Checkpoint 3 presentation, Sharpe ratio calculation, initial intraday data fetching.

* **Week of Nov 20: Checkpoint 4 Demo & Final Polish**

Objectives: Present Checkpoint 4, complete risk analysis and intraday charts, final documentation and testing. [9.5 hours]

Deliverables: Checkpoint 4 presentation, Sharpe ratio and introductory VaR calculation, 1-minute intraday charts (last 7 days, with limitations documented), completed Methods page.

# Timeline

Checkpoint #1 (Sept 25)

## Features: Home dashboard with ticker list, basic chart functionality with time ranges, navigation structure for all main pages, responsive design.

## Demo: Navigate home, swap tickers to update chart, demonstrate time range selection, show navigation to stub pages.

## Acceptance: Ticker navigation smooth, charts render correctly for time ranges, design responsive, nav links functional.

## Checkpoint #2 (Oct 9)

## Features: Watchlist CRUD, portfolio creation with inception dates, multi-year OHLCV candlesticks with volume pane, fundamentals (P/E, market cap).

## Demo: Create a watchlist, add tickers, create a portfolio, show ticker chart + volume + fundamentals.

## Acceptance: Watchlists persist, portfolios store inception, charts accurate, fundamentals display when available.

## Checkpoint #3 (Oct 30)

## Features: Paper trading with order placement + positions, portfolio analytics (CAGR, vol, max drawdown), SMA/EMA/RSI overlays, benchmark comparisons.

## Demo: Place a buy order and show updates, display analytics tiles, toggle indicators, add benchmark overlay.

## Acceptance: Trading updates positions/cash correctly, analytics calculations accurate, indicators render, benchmark overlays work, transaction history logs.

## Checkpoint #4 (Nov 20)

## Features: Risk metrics (Sharpe, preliminary VaR), 1-minute intraday charts (last 7 days), complete Methods page.

## Demo: Portfolio risk view with Sharpe/VaR, intraday tab with auto-refresh and update timestamp, Methods page with sources/formulas.

## Acceptance: Risk metrics calculated, intraday charts update correctly (with limitations noted), Methods page clear and complete, all features integrated smoothly.

# Future Enhancements

* **Expanded Asset Coverage**: Add support for ETFs beyond broad benchmarks, as well as basic fixed-income instruments, to demonstrate portfolio diversification.
* **Deeper Analytics**: Extend risk analysis to include advanced Value-at-Risk (VaR) and beta calculations, and allow users to compare portfolios against custom benchmarks.
* **Collaboration Features**: Let users share watchlists or “classroom portfolios” with others, supporting group projects or instructors who want to use the app in teaching.
* **Mobile Interface**: Adapt the responsive layout into a mobile-first design or standalone app for easier access outside the browser.
* **Improved Data Sources**: Replace or supplement yfinance with APIs offering more reliable intraday history and fundamentals, while keeping the platform free to use.
* **Learning Modules**: Integrate short, interactive lessons directly into the dashboard so users can practice a concept (like calculating returns) and immediately see it applied in their portfolio.

# Ethical Considerations

Access to accurate, transparent market information keeps getting harder as more networks and financial platforms move behind paywalls and costly subscriptions. Recent industry moves by CNN and Reuters to lock sites behind subscriptions illustrate a broader shift toward monetizing access, and surveys show most Americans simply stop reading when confronted with a paywall (Reuters, CNN Add Website Paywalls; Shearer). This creates barriers for learners and individual investors who lack institutional resources. Even regulators have noted these disparities: in 2020 the SEC overhauled the Market Data Infrastructure (Regulation NMS) to expand the amount of quote and trade information included in the core data feeds, aiming to reduce the advantage held by proprietary “premium” feeds (SEC Adopts Market Data Infrastructure Rules). Finvestor pushes in that same direction by acting as a free informational resource, explaining formulas, and making methods transparent, so beginners can learn without gatekeeping.

Paid promotions and stealth advertising add another ethical risk. Undisclosed sponsorships, influencer “tips,” and social-media hype can spread misinformation and foster false confidence in new investors. In the past, the SEC has issued investor alerts and enforcement actions against “pump-and-dump” (artificially inflating a commodity’s market value until it crashes) schemes and undisclosed touting, and the UK’s FCA has recently finalized guidance requiring that social-media promotions include fair, balanced, and prominent risk disclosure (*Financial Promotions on Social Media*). Finvestor takes the opposite stance: it will not accept paid placement or “pay-to-rank” behavior. All tickers receive equal treatment in charts and analytics, and any external content is labeled with source and date. This reinforces the principle that educational tools should inform, not manipulate.

Finally, Finvestor is explicitly a **simulation**—an educational sandbox, not investment advice. Clear disclaimers will state that simulated or back-tested performance is **not** indicative of future results, echoing long-standing disclosure requirements on hypothetical trading results (17 CFR § 4.41). Beyond formal warnings, the platform’s design itself emphasizes risk: use of volatility, drawdowns, Sharpe ratios, and other metrics is encouraged so users see that losses are as real as gains in market contexts. By highlighting risk and avoiding gamified leaderboards, Finvestor helps curb overconfidence that research has tied to misuse of hypothetical returns. The goal for now is a not-too-complex, transparent environment where beginners can practice safely. For long term, is to extend this ethos through open-source resources and possibly an educational nonprofit focused on personal finance, saving and investing, debt, and risk in under-resourced communities.

# Bibliography

* “17 CFR § 4.41 – Advertising by Commodity Pool Operators, Commodity Trading Advisors, and the Use of Hypothetical Performance Results.” *Code of Federal Regulations*, Commodity Futures Trading Commission, [https://www.law.cornell.edu/cfr/text/17/4.41](https://www.law.cornell.edu/cfr/text/17/4.41?utm_source=chatgpt.com). Accessed 31 Aug. 2025.
* Einhorn, Nick, et al. *The Retail Investor Report*. Public.com, Aug. 2023. UMKC School of Law Faculty Works. https://irlaw.umkc.edu/faculty\_works/928/. Accessed 31 Aug. 2025.
* *Financial Promotions on Social Media: Finalised Guidance FG24/1.* Financial Conduct Authority, Mar. 2024, [https://www.fca.org.uk/publication/finalised-guidance/fg24-1.pdf](https://www.fca.org.uk/publication/finalised-guidance/fg24-1.pdf?utm_source=chatgpt.com). Accessed 31 Aug. 2025.
* Gratton, Peter. “What Is a Bloomberg Terminal (BT)? Functions, Costs, and Inexpensive Alternatives.” *Investopedia*, 22 Nov. 2024. https://www.investopedia.com/terms/b/bloomberg\_terminal.asp. Accessed 31 Aug. 2025.
* Prachi. “Why Is It Important to Understand TimeSeries in Algo Trading?” *SpeedBot Blog*, 21 Aug. 2023. https://speedbot.tech/blog/algo-trading-4/understanding-timeseries-in-algo-trading-103. Accessed 31 Aug. 2025.
* “Reuters, CNN Add Website Paywalls.” *Axios*, 1 Oct. 2024, [https://www.axios.com/2024/10/01/reuters-cnn-add-website-paywalls. Accessed 31 Aug. 2025](https://www.axios.com/2024/10/01/reuters-cnn-add-website-paywalls.%20Accessed%2031%20Aug.%202025).
* “SEC Adopts Market Data Infrastructure Rules.” *U.S. Securities and Exchange Commission*, 9 Dec. 2020, https://www.sec.gov/news/press-release/2020-311. Accessed 31 Aug. 2025.
* Shearer, Elisa. “Few Americans Pay for News When They Encounter Paywalls.” *Pew Research Center*, 24 June 2025, [https://www.pewresearch.org/short-reads/2025/06/24/few-americans-pay-for-news-when-they-encounter-paywalls](https://www.pewresearch.org/short-reads/2025/06/24/few-americans-pay-for-news-when-they-encounter-paywalls?utm_source=chatgpt.com). Accessed 31 Aug. 2025.
* “Social Media and Investing: Avoiding Fraud.” *SEC Investor Alerts and Bulletins*, U.S. Securities and Exchange Commission, https://www.sec.gov/oiea/investor-alerts-bulletins/ia\_socialmedia. Accessed 31 Aug. 2025.